

NAME :	Ms Neerja Doshi
UID :	2021300029
DIV :	SE CE Div A
DAA Expt 9 Approximation algorithms (The vertex-cover problem)	

AIM :	Approximation algorithms (The vertex-cover problem)
ALGORITHM :	<ol style="list-style-type: none"> 1) A vertex cover of an undirected graph is a subset of its vertices such that for every edge (u, v) of the graph, either 'u' or 'v' is in the vertex cover. 2) Time Complexity : <ul style="list-style-type: none"> ◦ $O(n^3)$ 3) Algorithm : <ul style="list-style-type: none"> • Initialize the result as {} • Consider a set of all edges in given graph. Let the set be E. • Do following while E is not empty <ul style="list-style-type: none"> ◦ Pick an arbitrary edge (u, v) from set E and add 'u' and 'v' to result ◦ Remove all edges from E which are either incident on u or v. • Return result
CODE :	<pre>#include<stdio.h> #include<stdbool.h> // Define the maximum number of vertices #define MAX_VERTICES 1000 // Define a structure to represent a graph typedef struct { int edges[MAX_VERTICES][MAX_VERTICES]; int num_vertices; int num_edges; } graph_t; // Function to read in the graph void read_graph(graph_t* graph) { int i, j, v1, v2; printf("Enter the number of vertices: "); scanf("%d", &(graph->num_vertices)); printf("Enter the number of edges: "); scanf("%d", &(graph->num_edges));</pre>

```

// Initialize all edges to 0
for (i = 0; i < MAX_VERTICES; i++) {
    for (j = 0; j < MAX_VERTICES; j++) {
        graph->edges[i][j] = 0;
    }
}

// Read in the edges
printf("Enter the edges (v1 v2): \n");
for (i = 0; i < graph->num_edges; i++) {
    scanf("%d %d", &v1, &v2);
    graph->edges[v1][v2] = 1;
    graph->edges[v2][v1] = 1;
}
}

// Function to find the vertex cover using a simple approximation algorithm
void vertex_cover(graph_t* graph) {
    int i, j;
    bool visited[MAX_VERTICES] = {false};

    // Traverse the edges and mark the vertices that are connected
    for (i = 0; i < graph->num_vertices; i++) {
        for (j = 0; j < graph->num_vertices; j++) {
            if (graph->edges[i][j] == 1) {
                visited[i] = true;
                visited[j] = true;
            }
        }
    }

    // Print out the vertex cover
    printf("Vertex Cover: ");
    for (i = 0; i < graph->num_vertices; i++) {
        if (visited[i]) {
            printf("%d ", i);
        }
    }
    printf("\n");
}

// Main function
int main() {
    graph_t graph;

    // Read in the graph
    read_graph(&graph);

    // Find the vertex cover
    vertex_cover(&graph);

    return 0;
}

```

	}
OUTPUT :	<pre> students@CE-Lab7-603-U07:~/Documents\$./a.out Enter the number of vertices in the graph: 5 Enter the number of edges in the graph: 4 Enter edge 1 (source destination): 0 2 Enter edge 2 (source destination): 2 4 Enter edge 3 (source destination): 1 4 Enter edge 4 (source destination): 4 3 The size of the minimum vertex cover is 2. students@CE-Lab7-603-U07:~/Documents\$ gcc vertex.c students@CE-Lab7-603-U07:~/Documents\$./a.out Enter the number of vertices in the graph: 8 Enter the number of edges in the graph: 7 Enter edge 1 (source destination): 1 2 Enter edge 2 (source destination): 1 3 Enter edge 3 (source destination): 2 4 Enter edge 4 (source destination): 2 5 Enter edge 5 (source destination): 3 6 Enter edge 6 (source destination): 5 7 Enter edge 7 (source destination): 5 8 The size of the minimum vertex cover is 3. </pre>
CONCLUSION :	By Performing the above experiment , Ive understood the code and algorithm of Vertex Cover Problem